Gradient Weight-normalized Low-rank Projection for Efficient LLM Training

Jia-Hong Huang*, Yixian Shen*, Hongyi Zhu, Stevan Rudinac, Evangelos Kanoulas

University of Amsterdam
j.huang@uva.nl, y.shen@uva.nl, h.zhu@uva.nl, s.rudinac@uva.nl, E.Kanoulas@uva.nl

Abstract

Large Language Models (LLMs) have shown remarkable performance across various tasks, but the escalating demands on computational resources pose significant challenges, particularly in the extensive utilization of full fine-tuning for downstream tasks. To address this, parameter-efficient fine-tuning (PEFT) methods have been developed, but they often underperform compared to full fine-tuning and struggle with memory efficiency. In this work, we introduce Gradient Weight-Normalized Low-Rank Projection (GradNormLoRP), a novel approach that enhances both parameter and memory efficiency while maintaining comparable performance to full fine-tuning. GradNormLoRP normalizes the weight matrix to improve gradient conditioning, facilitating better convergence during optimization. Additionally, it applies low-rank approximations to the weight and gradient matrices, significantly reducing memory usage during training. Extensive experiments demonstrate that our 8-bit GradNormLoRP reduces optimizer memory usage by up to 89.5% and enables the pre-training of large LLMs, such as LLaMA 7B, on consumer-level GPUs like the NVIDIA RTX 4090, without additional inference costs. Moreover, GradNormLoRP outperforms existing low-rank methods in fine-tuning tasks. For instance, when fine-tuning the RoBERTa model on all GLUE tasks with a rank of 8, GradNormLoRP achieves an average score of 80.65, surpassing LoRA's score of 79.23. These results underscore GradNormLoRP as a promising alternative for efficient LLM pre-training and fine-tuning.

Code — https://github.com/Jhhuangkay/Gradient-Weight-normalized-Low-rank-Projection-for-Efficient-LLM-Training

Extended Version — https://arxiv.org/pdf/2412.19616

Introduction

Large Language Models (LLMs) pre-trained on extensive datasets have demonstrated exceptional effectiveness across various domains (Devlin et al. 2019; Liu et al. 2019; He et al. 2022; Xie et al. 2022; Baevski et al. 2020; Lu et al. 2019; Tan and Bansal 2019). As time progresses, open-source LLMs have consistently improved in their capabilities, accompanied by a striking increase in the scale of pre-trained models (Raffel et al. 2020a; Zhang et al. 2022; Le Scao et al. 2023; Touvron et al. 2023; Tay et al. 2023). Consequently, employing full fine-tuning, where all learnable parameters of a pre-trained model are updated for performing downstream tasks, poses unparalleled challenges despite its track record of delivering numerous state-of-the-art results. These challenges primarily stem from the escalating demands on computational resources.

Algorithm 1: Our proposed GradNormLoRP

Require: Weight matrix $\mathcal{W}$
Ensure: Updated weight matrix $\mathcal{W}_{\text{updated}}$
1: Normalize each column weight vector of $\mathcal{W}$ to get $\mathcal{W}_{\text{norm}}$
2: Apply LoRA with two low-rank matrices I and J to reformulate $\mathcal{W}_{\text{norm}}$
3: Initialize two sets of low-rank projection matrices (U_I, V_I) and (U_J, V_J)
4: **for** $i = 1$ to N **do**
5: Compute gradient matrices $\mathcal{Z}_I$ and $\mathcal{H}_J$ based on I and J
6: Project $\mathcal{Z}_I$ and $\mathcal{H}_J$ using (U_I, V_I) and (U_J, V_J)
7: **if** i is a multiple of 250 **then**
8: Update (U_I, V_I) and (U_J, V_J)
9: **end if**
10: **end for**
11: **return** $\mathcal{W}_{\text{updated}}(\mathcal{W}_{\text{norm}}, \mathcal{Z}_I, (U_I, V_I), \mathcal{H}_J, (U_J, V_J))$

To tackle the aforementioned challenge, researchers have developed parameter-efficient fine-tuning (PEFT) techniques (Houlsby et al. 2019; Hu et al. 2022; Lialin et al. 2023; Liu et al. 2024; Kopiczko, Blankevoort, and Asano 2024). These methods are tailored to update only a small amount of task-specific parameters while leaving the majority of the model's parameters unchanged. Among these techniques, low-rank approximation-based approaches utilize low-rank matrices to approximate weight changes during training, achieving both parameter and memory efficiency without requiring additional trainable subnetworks to be added to the original model architecture.

Despite their advantages, low-rank-based methods often underperform compared to full-rank fine-tuning (Hu et al. 2022; Lialin et al. 2023; Liu et al. 2024). This performance gap is typically attributed to the reduced number of trainable parameters, but other underlying factors, such as altered gradient dynamics due to reparameterization, also play a significant role. In Figure 2 of our **Extended Version**, we observe that the gradient descent process can become neither smooth nor stable when fine-tuning LLMs in an unnormalized subspace. This instability arises from conducting gradi-

*These authors contributed equally.

ent descent on an incomparable scale, where some values are excessively large or small. Such numerical instability can lead to overflow or underflow during computations, negatively impacting the optimization process and resulting in suboptimal performance. To mitigate this problem, we propose our method, Gradient Weight-Normalized Low-Rank Projection (GradNormLoRP). This approach effectively enhances both parameter and memory efficiency. GradNormLoRP improves parameter efficiency by incorporating a weight matrix normalization process that represents each column vector of the weight matrix as the product of its magnitude and unit vector. This normalization enhances gradient conditioning and facilitates better convergence during optimization.

In addition to improving parameter efficiency, GradNormLoRP addresses memory efficiency while maintaining performance comparable to full fine-tuning without introducing additional inference burden. Although training LLMs in a normalized subspace enhances convergence during optimization, existing PEFT methods (Hu et al. 2022; Liu et al. 2024; Houlsby et al. 2019; Pfeiffer et al. 2020; Li and Liang 2021; Lester, Al-Rfou, and Constant 2021; Liu et al. 2022) still face limitations in reducing GPU memory usage. Specifically, these methods rely on caching intermediate activations during the forward pass to compute gradients, which remains a significant memory overhead due to the standard backpropagation process. This inefficiency poses difficulties for training LLMs on a single consumer-level GPU, such as the NVIDIA RTX 4090 with 24GB of memory. To address the memory efficiency issue, GradNormLoRP applies a low-rank approximation technique to both the normalized weight matrix and its corresponding gradient matrix. This process involves reformulating the normalized weight matrix as the sum of a fixed pre-trained weight matrix and the product of two low-rank matrices. It also requires computing two sets of low-rank projection matrices to project the gradient matrices derived from these low-rank matrices. These projection matrices are updated periodically, e.g., every 250 iterations, to ensure minimal additional computational overhead over time. Exploiting this technique, our proposed GradNormLoRP achieves both memory and parameter efficiency during training while further enhancing the convergence process of optimization in the normalized subspace.

We conduct extensive experiments to demonstrate the effectiveness of our proposed GradNormLoRP in both LLM pre-training and fine-tuning, leveraging the C4 dataset (Raffel et al. 2020b) and the GLUE benchmark (Wang et al. 2019). GradNormLoRP significantly reduces memory usage in optimizer states by up to 89.5%, while preserving efficiency and performance during pre-training on the LLaMA 7B (Touvron et al. 2023) architecture with the C4 dataset, comprising up to 10.2 billion tokens. Furthermore, our 8-bit GradNormLoRP achieves additional reductions, cutting optimizer memory by up to 83.7% and total training memory by 65.2% compared to a BF16 baseline. Remarkably, we demonstrate the feasibility of pre-training the LLaMA 7B model on consumer-level GPUs with 24GB memory, such as the NVIDIA RTX 4090, without necessitating strategies like model parallelism, offloading, or checkpointing. In the realm of fine-tuning pre-trained LLMs on GLUE benchmarks, GradNormLoRP proves superior to existing low-rank methods. For example, when fine-tuning the RoBERTaBase model (Liu et al. 2019) on GLUE tasks with a rank of 8, GradNormLoRP attains an average score of 80.65, outpacing LoRA, which achieves a score of 79.23. The effectiveness of GradNormLoRP is also mathematically proved by our proposed **Theorem 1**. This highlights GradNormLoRP as a promising alternative to established methodologies within the field. The main contributions of this paper are as follows:

- **Development of GradNormLoRP for Enhanced LLM Training**: We introduce GradNormLoRP, a novel method designed to improve parameter and memory efficiency during the pre-training and fine-tuning of LLMs. GradNormLoRP enhances gradient conditioning, leading to better convergence during optimization while maintaining performance comparable to full fine-tuning.
- **Memory Efficiency on Consumer-Level GPUs**: GradNormLoRP addresses the memory efficiency limitations of existing PEFT methods. Through the application of low-rank approximation to both normalized weight matrices and their corresponding gradient matrices, the method substantially reduces memory usage in optimizer states, enabling the training of LLMs on consumer-level GPUs without the need for advanced memory management strategies.
- **Empirical and Theoretical Validation of GradNormLoRP's Effectiveness**: The effectiveness of GradNormLoRP is demonstrated through both theoretical analysis and extensive experimental evaluation. We provide a mathematical proof of GradNormLoRP's effectiveness, further solidifying its potential as a promising alternative to traditional fine-tuning approaches in the LLM domain.

Related Work

Parameter-Efficient Fine-Tuning. Numerous PEFT methods have emerged to address the computational challenges of fully fine-tuning LLMs. These methods can be grouped into those that increase model complexity and those that maintain or minimally modify the initial architecture. The first group, including methods like (Liao, Tan, and Monz 2023; Zhao et al. 2024a; Houlsby et al. 2019; Rebuffi, Bilen, and Vedaldi 2017; Gomez et al. 2017a; Pfeiffer et al. 2020; Rücklé et al. 2020; Li and Liang 2021; Lester, Al-Rfou, and Constant 2021; Hambardzumyan, Khachatrian, and May 2021; Liu et al. 2023), often incorporate trainable adapter layers or optimize input layer activations, which can add inference latency and pose challenges in large-scale, latency-sensitive environments. The second group of methods, including (Liu et al. 2024; Hu et al. 2022; Lialin et al. 2023), utilizes low-rank matrices to approximate weight changes during training. These low-rank matrices are designed to integrate seamlessly with pre-trained weights before inference, ensuring that no additional inference overhead is introduced. Our proposed GradNormLoRP belongs to this second category, leveraging the advantages of low-rank approximation methods without introducing extra inference latency.

Gradient Projection. Gradient projection is used for rapid low-rank estimation (Chen and Wainwright 2015; Chen, Raskutti, and Yuan 2019; Zhao et al. 2024b). The work in (Chen and Wainwright 2015; Chen, Raskutti, and Yuan 2019) treats the objective function as a general non-linear function, analyzing gradients in vector space. GaLore (Zhao et al. 2024b), however, considers the specific structures of gradients in multi-layer neural networks, establishing that gradients tend to become low-rank during training and exhibit specific convergence behaviors. Further studies (Larsen et al. 2022; Gur-Ari, Roberts, and Dyer 2018) demonstrate that effective learning often takes place in a low-dimensional subspace, optimizing model weights within this constrained space—a process known as subspace learning. Our proposed

GradNormLoRP advances this concept by operating within a low-dimensional normalized subspace, enhanced by weight matrix normalization.

GPU-Memory-Efficient Training. Several techniques have been developed to optimize GPU memory utilization during LLM training. Reversible subnetworks (Liao, Tan, and Monz 2023; Mangalam et al. 2022; Zhao et al. 2024a; Gomez et al. 2017b; Kitaev, Kaiser, and Levskaya 2020) minimize activation memory by recalculating activations during backpropagation. Gradient checkpointing (Chen et al. 2016) improves memory efficiency by discarding and later reconstructing some intermediate activations through an additional forward pass. Pruning (Frankle and Carbin 2019; Frankle et al. 2020) and knowledge distillation (Sanh et al. 2020; Hinton, Vinyals, and Dean 2015; Koratana et al. 2019) compress models by removing redundant parameters or transferring distilled knowledge. Using pre-trained models as feature extractors without gradient computation also reduces activation memory (Liu, An, and Qiu 2024; Sung, Cho, and Bansal 2022). Quantization reduces optimizer state memory overhead (Dettmers et al. 2022; Li, Chen, and Zhu 2023). Fused gradient calculation (Lv et al. 2023) alleviates memory overhead from storing weight gradients, and Adafactor (Shazeer and Stern 2018) reduces memory costs by factorizing second-order statistics. Unlike these approaches, GradNormLoRP provides optimizers with low-rank gradients directly, eliminating the need for full-rank gradient knowledge.

Methodology

In this section, we detail the key components of our GradNormLoRP and establish a theorem that theoretically demonstrates the effectiveness of GradNormLoRP in preserving the integrity of training dynamics. Please consult **Algorithm 1** for a more comprehensive grasp of our GradNormLoRP.

Background

Weight Vector Normalization. Weight vector normalization is a technique that can be employed to expedite the convergence of the stochastic gradient descent optimization process (Srebro and Shraibman 2005; Salimans and Kingma 2016). We consider standard neural networks in which each neuron's computation involves calculating a weighted sum of input features, followed by a component-wise non-linearity:

$$y = \theta((\sum_{i=1}^{k} w_i a_i) + b) = \theta(\langle w, a\rangle + b), \tag{1}$$

where $w \in \mathbb{R}^{k\times 1}$ represents a weight vector, $a \in \mathbb{R}^{k\times 1}$ signifies an input feature vector, $b \in \mathbb{R}$ indicates a bias term, $\langle\cdot,\cdot\rangle$ denotes the inner product, $\theta(\cdot)$ is an component-wise nonlinearity, e.g., the logistic activation $\frac{\exp(\cdot)}{1+\exp(\cdot)}$, and y indicates the scalar output of the neuron.

After a loss function is associated with one or more neuron outputs, the parameters w and b for each neuron are typically optimized using stochastic gradient descent during the training of such a neural network. To enhance the convergence of the optimization process, a reparameterization operation is introduced to express each weight vector w in terms of a parameter vector v and a scalar parameter δ:

$$w = \delta\frac{v}{\|v\|}, \tag{2}$$

where $\delta \in \mathbb{R}$ denotes a scalar, $v \in \mathbb{R}^{k\times 1}$, and $\|\cdot\|$ indicates the Euclidean norm.

This reparameterization, which decouples the weight vector's norm (δ) from the direction of the weight vector ($\frac{v}{\|v\|}$), fixes the Euclidean norm of the weight vector, yielding $\|w\| = \delta$, which remains independent of the parameter vector v. After employing the reparameterization weight normalization process, we obtain:

$$y = \theta(\langle w, a\rangle + b) = \theta(\langle \delta\frac{v}{\|v\|}, a\rangle + b). \tag{3}$$

Subsequently, the optimization process of stochastic gradient descent is conducted to the new parameters v and δ instead. In our proposed GradNormLoRP approach, we conduct the operation of reparameterization weight normalization on each column weight vector of a given weight matrix, resulting in a normalized weight matrix.

Challenges in Memory Efficiency for PEFT. As discussed in (Raffel et al. 2020a; Zhao et al. 2024b; Liao, Tan, and Monz 2023; Touvron et al. 2023), the primary memory consumption during neural network training is attributed to activations, trainable parameters, and gradients of these parameters, along with optimizer states such as gradient momentum and variance in Adam (Kingma and Ba 2017). In this subsection, we employ a T-layer multilayer perceptron to illustrate the main origin of the memory efficiency issue inherent in low-rank approximation-based PEFT methods. Consider a T-layer multilayer perceptron: $h_T = \xi_T(\xi_{T-1}(\ldots(\xi_2(\xi_1(h_0)))\ldots))$ with h_0 as the initial input, where the t^{th} layer $h_t = \xi_t(h_{t-1}) = \phi_t(W_t h_{t-1})$ comprises a nonlinear function ϕ_t and a weight matrix W_t, neglecting the bias term for simplicity. Let $\psi_t = W_t h_{t-1}$. During the process of backpropagation with a loss $\mathcal{L}$, the gradient of W_t is computed using the chain rule as:

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial W_t} &= \frac{\partial\mathcal{L}}{\partial h_T}\left(\prod_{i=t+1}^{T}\frac{\partial h_i}{\partial\psi_i}\frac{\partial\psi_i}{\partial h_{i-1}}\right)\frac{\partial h_t}{\partial\psi_t}\frac{\partial\psi_t}{\partial W_t}\\ &= \frac{\partial\mathcal{L}}{\partial h_T}\left(\prod_{i=t+1}^{T}\phi_i' W_i\right)\phi_t' h_{t-1},\end{aligned} \tag{4}$$

where $\frac{\partial h_i}{\partial\psi_i} = \phi_i'$, $\frac{\partial\psi_i}{\partial h_{i-1}} = W_i$, $\frac{\partial h_t}{\partial\psi_t} = \phi_t'$, and $\frac{\partial\psi_t}{\partial W_t} = h_{t-1}$. Since ϕ_t' represents the derivative of ϕ_t and the computation of ϕ_t' relies on ψ_t, caching the sequence of activations $\{\psi_i\}_{i=t}^{T}$ during the forward pass is essential to compute the gradient of W_t, even though $\{W_i\}_{i>t}$ remain frozen. In contrast to full fine-tuning, existing low-rank approximation-based PEFT methods adjust only a limited number of parameters, resulting in a negligible size of the optimizer state (Liu et al. 2024; Hu et al. 2022; Lialin et al. 2023; Kopiczko, Blankevoort, and Asano 2024). Nevertheless, there is no significant reduction in the memory consumption required for activations. Take $\text{BERT}_{\text{base}}$ fine-tuned on the RTE benchmark with a batch size of 64 and sequence length of 512: the PEFT methods still require over 75% of the activation memory used in full fine-tuning, even though their trainable parameters are reduced to less than 1% (Devlin et al. 2018; Liao, Tan, and Monz 2023; Bentivogli et al. 2009).

Our Proposed GradNormLoRP

Gradient Projection. The efficacy of existing low-rank approximation-based PEFT approaches, such as LoRA (Hu et al. 2022), often falls short in comparison to full fine-tuning,

primarily due to their limited number of trainable parameters and the potential change of gradient training dynamics resulting from the low-rank reparameterization process (Xia, Qin, and Hazan 2024; Zhao et al. 2024b; Kopiczko, Blankevoort, and Asano 2024). A promising avenue to mitigate this challenge is through gradient projection techniques (Chen and Wainwright 2015; Chen, Raskutti, and Yuan 2019; Zhao et al. 2024b). The core concept behind gradient projection is to leverage the gradual evolution of the low-rank structure within the gradient of a weight matrix, instead of directly approximating the weight matrix as done in the LoRA method. This principle is grounded on the claim that the gradient tends to exhibit low-rank characteristics as training progresses. In this subsection, we substantiate this claim through rigorous proof.

Weight Matrix Updates in Conventional Full-Rank Training. Given $\mathcal{D}_t = -\nabla_{\mathcal{W}}\mathcal{L}(\mathcal{W}_t) \in \mathbb{R}^{k\times m}$ as the representation of the backpropagated negative gradient matrix at time step t, the traditional pre-training weight update with a learning rate α can be expressed as follows:

$$\mathcal{W}_T = \mathcal{W}_0 + \alpha \sum_{t=0}^{T-1} \widetilde{\mathcal{D}}_t = \mathcal{W}_0 + \alpha \sum_{t=0}^{T-1} \eta_t(\mathcal{D}_t), \quad (5)$$

where $\widetilde{\mathcal{D}}_t$ represents the final processed gradient added to the weight matrix, and η_t denotes a component-wise stateful gradient regularizer, such as Adam.

Weight Matrix Updates in Low-Rank Approximation-Based Methods. For a linear layer with a weight matrix $\mathcal{W} \in \mathbb{R}^{k\times m}$, approaches, such as LoRA, which are based on low-rank approximation, leverage the low-rank structure of the update matrix by introducing a low-rank adaptor IJ.

$$\mathcal{W}_T = \mathcal{W}_0 + I_T J_T, \quad (6)$$

where $I \in \mathbb{R}^{k\times r}$, $J \in \mathbb{R}^{r\times m}$, and $r \ll \min(k, m)$. I and J denote the trainable low-rank adaptors, while $\mathcal{W}_0$ stands as a fixed weight matrix, such as a pre-trained weight matrix.

While low-rank updates are suggested to alleviate memory consumption, there is ongoing debate regarding whether the weight matrix should inherently adopt a low-rank parameterization. This assumption may not hold in various scenarios, such as linear regression. However, the gradient often exhibits low-rank characteristics during training, especially with specific gradient forms and associated network architectures (Zhao et al. 2024b). The proof for **Lemma 1** is available in our **Extended Version**.

Lemma 1 (Gradient Becoming Low-rank during Training). *Given $\mathcal{W}_t \in \mathbb{R}^{k\times m}$, where we assume $k \leq m$ without loss of generality. Consider the gradient matrix $\mathcal{D}_t = \mathcal{A} - \mathcal{B}\mathcal{W}_t\mathcal{C}$, where $\mathcal{A}$ denotes a constant matrix, $\mathcal{B}$ and $\mathcal{C}$ both are positive semidefinite (PSD) matrices, and $\mathcal{W}_0$ is randomly initialized. Then, the gradient in the update of weight matrix $\mathcal{W}_t = \mathcal{W}_{t-1} + \alpha\mathcal{D}_{t-1}$ results in low-rank gradient with high probability:*

$$\text{stable-rank}(\mathcal{D}_t) \leq 1 + \sum_{i=2}^{k} O\left(\left(\frac{1-\alpha\lambda_i\nu_1}{1-\alpha\lambda_1\nu_1}\right)^{2t}\right), \quad (7)$$

where $\nu_1 = \lambda_{\min}(\mathcal{C})$ is the smallest eigenvalue of $\mathcal{C}$ and $\lambda_1 \leq \ldots \leq \lambda_m$ are eigenvalues of $\mathcal{B}$. Moreover, if $\mathcal{C}$ is positive definite, i.e., $\nu_1 > 0$, and $\lambda_2 > \lambda_1$, $\mathcal{D}_t$ converges exponentially to rank-1.

Normalization of Weight Matrix. The initial phase of our proposed GradNormLoRP involves normalizing a provided weight matrix $\mathcal{W}$. This normalization entails reparameterizing each column vector of the weight matrix using the operation introduced in section *"Weight Vector Normalization"*. The normalization process of the weight matrix $\mathcal{W} \in \mathbb{R}^{k\times m}$ can be expressed as follows:

$$\mathcal{W} = ||\mathcal{W}||_c \frac{\mathcal{W}}{||\mathcal{W}||_c} = \mathcal{M}\frac{\mathcal{W}}{||\mathcal{W}||_c}, \quad (8)$$

where $\mathcal{M} \in \mathbb{R}^{1\times m}$ indicates the reparameterized, i.e., trainable, length vector, $\mathcal{W}/||\mathcal{W}||_c \in \mathbb{R}^{k\times m}$ represents the directional matrix, and $||\cdot||_c$ denotes the vector-wise matrix norm operated across each column.

After performing the reparameterization weight normalization operation column-wise on the weight matrix, we have disentangled the magnitude of the weight vectors from their direction. This process ensures that each column of $\mathcal{W}/||\mathcal{W}||_c$ becomes a unit vector with an associated scalar. Each scalar element in vector $\mathcal{M}$ represents the length of a corresponding vector in weight matrix $\mathcal{W}$.

Low-rank Approximation. The proposed GradNormLoRP is initialized with pre-trained weight $\mathcal{W}_0$ as shown in Equation (8), where $\mathcal{M} = ||\mathcal{W}_0||_c$ and $\mathcal{W} = \mathcal{W}_0$ after initialization. Subsequently, we freeze $\mathcal{W}$ while making $\mathcal{M}$ serve as a trainable vector. The directional matrix is then updated using low-rank approximation techniques, such as LoRA. GradNormLoRP can be formulated similarly to Equation (6) as follows:

$$\mathcal{W} = \mathcal{M}\frac{\mathcal{W}_0 + IJ}{||\mathcal{W}_0 + IJ||_c}, \quad (9)$$

where $\mathcal{M}$ represents a vector comprising trainable parameters, while the weight matrices $I \in \mathbb{R}^{k\times r}$ and $J \in \mathbb{R}^{r\times m}$ are initialized following LoRA's approach to guarantee that $\mathcal{W}$ equals $\mathcal{W}_0$ before fine-tuning.

As the introduced low-rank approximation in our GradNormLoRP can be merged with the pre-trained weight before inference, it does not introduce any additional latency in the inference phase.

Gradient Projection Process. To enhance the convergence of the optimization process while simultaneously reducing memory usage during training, we integrate the gradient projection technique introduced in section *"Gradient Projection"* into our proposed GradNormLoRP.

Singular Value Decomposition (SVD) and Projection Matrices. In this study, we utilize SVD to obtain projection matrices that serve the purpose of gradient projection for the gradient matrix $\mathcal{D}_t$:

$$\mathcal{D}_t = USV^\top \approx \sum_{i=1}^{r} s_i u_i v_i^\top. \quad (10)$$

Let $\mathcal{U}_t = [u_1, u_2, \ldots, u_r]$ and $\mathcal{V}_t = [v_1, v_2, \ldots, v_r]$ denote projection matrices. Then, $\tilde{\mathcal{D}}_t$ in Equation (5) can be expressed as follows:

$$\tilde{\mathcal{D}}_t = \mathcal{U}_t\eta_t(\mathcal{U}_t^\top\mathcal{D}_t\mathcal{V}_t)\mathcal{V}_t^\top. \quad (11)$$

As per **Lemma 1** , the gradient $\mathcal{D}$ may exhibit a low-rank structure. Therefore, by preserving the gradient statistics of a compact "key portion" of $\mathcal{D}$ in optimizer states instead of $\mathcal{D}$ itself, significant reductions in memory consumption can be achieved. This motivates the gradient projection strategy integrated into our proposed GradNormLoRP.

Definition 1 (Gradient Projection in GradNormLoRP). *The gradient projection strategy in our proposed GradNormLoRP, with a learning rate α, follows these gradient update rules:*

$$\mathcal{W}_t = \mathcal{W}_0 + \alpha \sum_{t=0}^{T-1} \tilde{\mathcal{Z}}_t \tilde{\mathcal{H}}_t, \tilde{\mathcal{Z}}_t = P_t \eta_t (P_t^\top \mathcal{Z}_t Q_t) Q_t^\top,$$
$$\tilde{\mathcal{H}}_t = \mathcal{P}_t \eta_t (\mathcal{P}_t^\top \mathcal{H}_t \mathcal{Q}_t) \mathcal{Q}_t^\top, \quad (12)$$

where $\mathcal{W}_0 \in \mathbb{R}^{k \times m}$ denotes the initial weight matrix; $\mathcal{Z}_t \in \mathbb{R}^{k \times r}$ and $\mathcal{H}_t \in \mathbb{R}^{r \times m}$ are the low-rank gradient matrices of the weight matrices I_t and J_t in Equation (9), respectively. $P_t \in \mathbb{R}^{k \times r}$, $Q_t \in \mathbb{R}^{r \times m}$, $\mathcal{P}_t \in \mathbb{R}^{r \times h}$, and $\mathcal{Q}_t \in \mathbb{R}^{m \times s}$ are projection matrices.

In contrast to LoRA, our proposed GradNormLoRP adopts a distinct approach by employing two low-rank updates, i.e., $\tilde{\mathcal{Z}}_t$ and $\tilde{\mathcal{H}}_t$, explicitly, avoiding the introduction of additional low-rank adaptors and thereby mitigating the alteration of training dynamics. Essentially, integrating GradNormLoRP into model training facilitates smooth transitions across normalized low-rank subspaces, as delineated in Equation (13). This means the model can smoothly navigate between different sets of parameters, akin to switching lanes on a highway to optimize its learning process.

$$\mathcal{W}_t = \mathcal{W}_0 + \Delta\mathcal{W}_{T_1} + \Delta\mathcal{W}_{T_2} + \ldots + \Delta\mathcal{W}_{T_m}, \quad (13)$$

where $t \in [\sum_{i=1}^{m-1} T_i, \sum_{i=1}^{m} T_i]$ and $\Delta\mathcal{W}_{T_i} = \alpha \sum_{t=0}^{T_i-1} \tilde{\mathcal{D}}_t$ denotes the sum of all T_i updates within the i-th normalized subspace.

The effectiveness of GradNormLoRP hinges on the premise that gradients often exhibit low-rank properties throughout training. To validate this assertion, we present **Theorem 1** , with its proof provided in our **Extended Version**.

Theorem 1 *Let $r \leq m$ without loss of generality. The gradient update rules of GradNormLoRP:*

$$\mathcal{Z}_t = A - BI_tC,\ I_t = I_{t-1} + \gamma\mathcal{Z}_{t-1},$$
$$\mathcal{H}_t = E - FJ_tG,\ J_t = J_{t-1} + \beta\mathcal{H}_{t-1}, \quad (14)$$

with constant matrices (A and E), PSD matrices (B, C, F, and G), and randomly initialized I_0 and J_0 leads to low-rank gradient with high probability:

$$\text{stable-rank}(\mathcal{Z}_t, \mathcal{H}_t) \leq 1 + \sum_{i=2}^{r} O\left(\left(\frac{1-\gamma\omega_i\nu_1}{1-\gamma\omega_1\nu_1}\right)^{2t}\right) \times \sum_{j=2}^{r} O\left(\left(\frac{1-\beta\pi_j\mu_1}{1-\beta\pi_1\mu_1}\right)^{2t}\right). \quad (15)$$

Experiments

In this section, we evaluate the efficacy of our proposed GradNormLoRP through a series of experiments. We assess its performance in fine-tuning and pre-training scenarios and conduct a thorough throughput analysis to confirm that GradNormLoRP integrates seamlessly without adding inference latency. Additionally, we perform comprehensive ablation studies to highlight GradNormLoRP's characteristics, including convergence speed, parameter efficiency, and GPU memory utilization.

Experimental Setup

Datasets, Evaluation Metrics, Model Architectures, and Baselines. For fine-tuning, we use the GLUE benchmark (Wang et al. 2019), which includes single-sentence tasks (CoLA, SST-2), similarity and paraphrase tasks (MRPC, QQP, STS-B), and inference tasks (MNLI, QNLI, RTE, WNLI). Evaluation metrics are accuracy for MNLI, QQP, QNLI, SST-2, MRPC, and RTE, Pearson and Spearman correlation for STS-B, and Matthews correlation for CoLA. For pre-training, we use the C4 dataset (Raffel et al. 2020b), a cleaned version of Common Crawl's web corpus, with perplexity as the performance metric.

In our fine-tuning experiments, we use BERTbase (Devlin et al. 2018), RoBERTabase, RoBERTalarge (Liu et al. 2019), and BARTbase (Lewis et al. 2019) for all GLUE tasks. For pre-training, we adopt the LLaMA model architecture, training on the C4 dataset with no data repetition, scaling up to 7 billion parameters.

Our primary baseline for fine-tuning is full parameter updating. For PEFT experiments, we compare GradNormLoRP against LoRA (Hu et al. 2022), DoRA (Liu et al. 2024), and GaLore (Zhao et al. 2024b), which are also used as baselines for our pre-training experiments due to their relevance in low-rank approximation methods.

Implementation. For fine-tuning, we evaluated our model on the GLUE benchmark, exploring learning rates in the range of {1e-4, 2e-4, 3e-4, 4e-4, 5e-4}, batch sizes of 16 and 32, and a fixed number of 30 epochs. Specifically, we used a batch size of 16 for all tasks except for CoLA, which used a batch size of 32. The maximum sequence length for all tasks was set to 512 for $\text{BERT}_{\text{base}}$, $\text{RoBERTa}_{\text{base}}$, $\text{RoBERTa}_{\text{large}}$, and $\text{BART}_{\text{base}}$ models. For pretraining, we applied GradNormLoRP across various model sizes ranging from 60M to 1B parameters. The hyperparameters for GradNormLoRP were consistent across all models, with a learning rate of 0.01 and a scale factor (α_s) of 0.25. The learning rate was fine-tuned from the set {1e-2, 1e-3,5e-4,1e-4}, selecting the best rate based on validation perplexity. Each model was pre-trained for 10,000 steps. For models scaled up to 7B parameters, we set the batch size to 16 and varied the training steps accordingly.

Results and Analysis

Quantitative Results for Fine-tuning. We fine-tune pre-trained RoBERTa models on GLUE tasks using GradNormLoRP and compare its performance with a full fine-tuning baseline, LoRA, DoRA, and GaLore. We use hyperparameters from (Hu et al. 2022) for LoRA and (Liu et al. 2024) for DoRA, and tune the learning rate and scale factor for GradNormLoRP. As shown in Table 1, GradNormLoRP achieves better performance than LoRA and DoRA on most tasks with a lower memory footprint. For instance, GradNormLoRP achieves an average score of 81.01 with a memory usage of 249M for rank=4, while LoRA and DoRA achieve average scores of 80.40 and 80.52 with memory usages of 257M and 259M, respectively. This demonstrates that GradNormLoRP can serve as a full-stack memory-efficient training strategy for fine-tuning. Specifically, GradNormLoRP shows notable improvements in tasks such as SST-2, where it achieves 94.50 compared to 92.89 for GaLore. Additionally, GradNormLoRP maintains competitive performance in other tasks like QNLI and QQP, demonstrating its robustness.

Quantitative Results for Pre-training. For GradNormLoRP and GaLore, we set subspace frequency T to 250 and scale factor α_s to 0.25 across all model sizes in Table 2. For each model size, we pick the same rank r for all low-rank methods,

Model	Memory	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	WNLI	Avg
Full FT	747M	57.03	86.30	92.09	92.04	91.35	77.62	91.74	90.82	43.90	80.32
LoRA (r=4)	257M	55.27	86.81	89.95	92.42	89.44	69.68	93.58	90.07	**56.34**	80.40
DoRA (r=4)	259M	55.51	**86.90**	89.91	92.24	89.54	70.75	93.46	90.04	**56.34**	80.52
GaLore (r=4)	253M	**60.65**	85.65	**91.14**	90.76	**90.70**	**77.26**	92.89	90.84	36.62	79.61
Ours (r=4)	249M	59.31	86.42	91.10	**92.48**	90.60	75.81	**94.50**	**90.85**	47.89	**81.01**
LoRA (r=8)	264M	56.50	86.65	90.30	92.60	89.74	69.68	93.81	90.10	**43.67**	79.23
DoRA (r=8)	267M	57.27	86.55	89.56	92.46	89.86	70.04	94.15	90.16	43.66	79.30
GaLore (r=8)	257M	52.59	85.66	**92.06**	91.31	90.74	**78.70**	92.66	90.80	39.44	79.33
Ours (r=8)	251M	**60.31**	**86.97**	91.36	**92.62**	**91.01**	77.26	**94.50**	**90.83**	40.85	**80.65**

Table 1: Evaluating GradNormLoRP for memory-efficient fine-tuning on the GLUE benchmark using the pre-trained $\text{RoBERTa}_{\text{base}}$ model. "r" indicates rank, "Ours" signifies GradNormLoRP, and "FT" denotes fine-tuning.

Method	60M	130M	350M	1B
Full-rank	34.51;(0.35)	25.91;(0.8)	20.24;(2.21)	16.86;(8.03)
LoRA	35.33;(0.35)	30.55;(0.81)	25.11;(1.93)	22.35;(6.32)
DoRA	35.42;(0.37)	30.92;(0.82)	24.91;(1.95)	21.98;(6.37)
GaLore	34.94;(0.23)	26.57;(0.54)	20.64;(1.47)	16.77;(4.69)
Ours	**34.63**;(0.17)	**26.52**;(0.47)	**19.28**;(1.19)	**16.12**;(3.81)
r/d_{model}	128 / 256	256 / 768	256 / 1024	512 / 2048
Training Tokens	1.1B	2.2B	6.4B	13.1B

Table 2: Compared with low-rank algorithms on pre-training various sizes of LLaMA models on the C4 dataset, reporting validation perplexity and memory estimates for parameters and optimizer states in BF16 format, with actual memory usage as shown. Note that the unit in each parenthesis is "G".

Model	Memory	20K	40K	60K	80K
8-bit GradNormLoRP	**15.29G**	**19.33**	**17.73**	**16.43**	**15.41**
8-bit GaLore	18.36G	20.19	18.15	16.96	16.08
8-bit Adam	26.47G	20.65	18.31	17.11	16.24
Training Tokens		2.5B	5.2B	7.7B	10.5B

Table 3: Pre-training LLaMA 7B on the C4 dataset for 80K steps, with validation perplexity and memory estimates reported.

applying them to all the linear layers of all multi-head attention layers and feed-forward layers in the models. We keep the Adam optimizer settings consistent with GaLore (Zhao et al. 2024b). We also estimate the memory usage based on BF16 format, including the memory for weight parameters and optimizer states. We evaluate them on LLaMA 60M, 130M, 350M and 1B architecture with 10K training steps, and we tune the learning rate for each setting and report the best performance.

As shown in Table 2, GradNormLoRP achieves significant reductions in validation perplexity and memory usage across different model sizes compared to other methods. For instance, for the 350M parameter model, GradNormLoRP achieves a perplexity of 19.28 with a memory usage of 1.19G, whereas GaLore achieves a perplexity of 20.64 with a memory usage of 1.47G. This represents a substantial improvement in both memory efficiency and model performance. Similarly, for the 1B parameter model, GradNormLoRP reduces the perplexity to 16.12 and memory usage to 3.81G, outperforming GaLore's perplexity of 16.77 and memory usage of 4.69G. These results demonstrate that GradNormLoRP can significantly enhance the efficiency of pre-training LLMs, making it a robust and scalable solution for training in resource-constrained environments.

Pre-training on LLaMA 7B. Scaling to 7B models is crucial for demonstrating GradNormLoRP's effectiveness in practical LLM pre-training. We evaluate GradNormLoRP on an LLaMA 7B architecture, which has an embedding size of 4096 and 32 layers. The model is trained for 80K steps with 10.5B tokens, using 8-node parallel training on 32 A100 GPUs. Due to computational constraints, we compare 8-bit GradNormLoRP (r = 1024) with 8-bit Adam, performing a single trial without hyperparameter tuning. As shown in Table 3, 8-bit GradNormLoRP not only has a lower memory footprint but also achieves better performance metrics. Specifically, 8-bit GradNormLoRP requires 15.29GB of memory, significantly less than the 18.36GB and 26.47GB required by 8-bit GaLore and 8-bit Adam, respectively.

In terms of validation perplexity, 8-bit GradNormLoRP consistently outperforms other methods across training steps, achieving lower perplexity with higher ranks. At 20K steps, 8-bit GradNormLoRP reaches a perplexity of 19.33, compared to 20.19 for 8-bit GaLore and 20.65 for 8-bit Adam. This trend persists at 40K, 60K, and 80K steps, where 8-bit GradNormLoRP achieves 15.41 perplexity, outperforming 8-bit GaLore (16.08) and 8-bit Adam (16.24). Its superior performance stems from efficient low-rank adaptation during gradient projection, optimizing memory usage and enhancing training efficiency, making it a highly effective solution for pre-training LLMs with improved hardware utilization.

Regarding memory consumption, as shown in the left subfigure of Figure 1, 8-bit GradNormLoRP requires only 20.07GB to pre-train the LLaMA 7B model with a per-GPU token batch size of 256, well within the 24GB VRAM of an NVIDIA RTX 4090. This is substantially lower than BF16 and 8-bit Adam, which exceed 24GB for larger models.

Subspace Update Frequency Ablation Study. This ablation study, illustrated in the middle subfigure of Figure 1, highlights the importance of finding an optimal update frequency for subspace updates to achieve the best model convergence. Both overly frequent and overly infrequent updates negatively impact performance, leading to increased perplexity. The study shows that the optimal performance occurs at a moderate update frequency of around 250 iterations, especially for higher ranks such as 256 and 512, which benefit from more effective optimization within larger subspaces. Weight normalization plays a crucial role by stabilizing the gradient descent process, ensuring that updates are on a comparable scale and preventing inefficiencies.

Rank of Subspace Ablation Study. As shown in the right subfigure of Figure 1, this study examines how subspace rank and training steps affect perplexity. Training with rank 128 for 80K steps achieves better perplexity than rank 512 at 20K steps, emphasizing the importance of sufficient training duration for higher ranks. GradNormLoRP excels by effectively combining subspace updates and weight normalization, stabi-

Model	Memory	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	WNLI	Avg
Full $FT_{RoBERTa}$	747M	57.03	86.30	92.09	92.04	91.35	77.62	91.74	90.82	43.90	80.32
Full FT_{BART}	901M	53.78	86.16	91.66	91.83	91.01	77.11	91.75	89.87	39.82	79.25
Full FT_{BERT}	708M	56.47	86.29	91.72	91.97	91.04	77.12	91.79	89.99	39.85	79.60
$Ours_{RoBERTa}$	251M	**60.31**	**86.97**	91.36	**92.62**	**91.01**	77.26	**94.50**	**90.83**	40.85	**80.65**
$Ours_{BART}$	361M	54.13	86.29	91.00	91.85	90.07	73.31	93.23	89.15	49.30	79.81
$Ours_{BERT}$	236M	58.57	86.95	89.75	90.94	91.42	70.76	92.20	88.69	44.99	79.26

Table 4: Model architecture ablation study under the same model size. "r" denotes rank, "Ours" indicates GradNormLoRP, and "FT" signifies fine-tuning. GradNormLoRP utilizes a rank of 8 here.

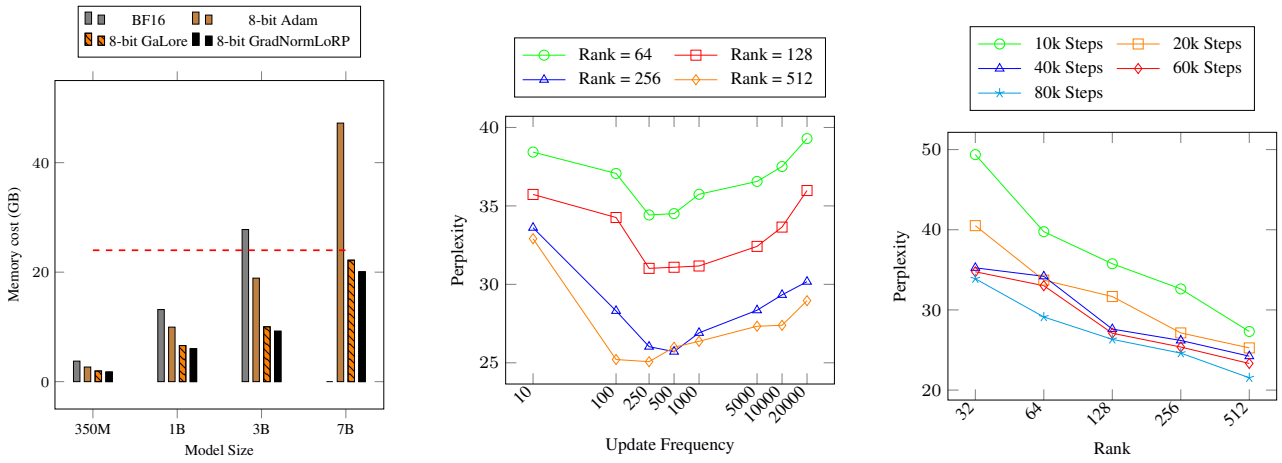

Figure 1: From left to right, the figure illustrates a comparison of memory usage, the impact of varying subspace frequencies, and the effect of rank across steps. Note that the red dashed line denotes the RTX 4090.

lizing gradient descent. While higher ranks offer significant gains, they require more steps, whereas lower ranks optimize efficiently with fewer steps but show more gradual perplexity improvements.

Model Architecture Ablation Study. We evaluated GradNormLoRP on BARTbase, BERTbase, and RoBERTabase to assess its robustness. As shown in Table 4, GradNormLoRP consistently outperforms full fine-tuning. For RoBERTabase at rank=8, it achieves an average score of 80.65, with significant improvements in CoLA (60.31) and SST-2 (94.50), compared to 80.32 for full fine-tuning. For BARTbase, it scores 79.81 on average, excelling in MNLI (86.97) and MRPC (91.06) over full fine-tuning's 79.25. For BERTbase, it maintains a competitive average score of 79.26, with notable results in QNLI (89.75) and QQP (91.42), compared to 79.60 for full fine-tuning.

Model Size Ablation Study. We evaluated GradNormLoRP on RoBERTabase and RoBERTalarge, demonstrating significant memory savings while maintaining or improving performance compared to full fine-tuning (see Table 5 in the **Extended Version**). For RoBERTabase (rank=8), GradNormLoRP achieves an average score of 80.63, with notable gains in CoLA (60.31 vs. 57.03) and SST-2 (94.50 vs. 91.74). For RoBERTalarge, it achieves 82.43, slightly outperforming full fine-tuning (82.39), underscoring GradNormLoRP's efficiency across model sizes.

Weight Normalization Ablation Study. We conducted an ablation study to evaluate the impact of weight normalization on the $RoBERTa_{base}$ model's performance across GLUE benchmark tasks. As shown in Table 6 in our **Extended Version**, weight normalization consistently improves performance. The improvement is most noticeable in tasks like CoLA, where accuracy significantly increases. While gains in SST-2 are minor, they align with the overall positive trend. In more complex tasks like MRPC, weight normalization provides notable benefits, indicating its role in stabilizing and optimizing the training process.

Gradient Projection Ablation Study. We compared the performance and memory usage of the $RoBERTa_{base}$ model with and without gradient projection across GLUE tasks. As shown in Table 7 in our **Extended Version**, incorporating gradient projection in GradNormLoRP significantly boosts performance while preserving memory efficiency. Models with gradient projection demonstrate improved stability and training dynamics, leading to higher scores in tasks like CoLA and SST-2.

Conclusion

GradNormLoRP addresses the growing computational demands of full fine-tuning for LLMs by enhancing parameter and memory efficiency while maintaining comparable performance. By normalizing weight matrices, applying low-rank approximation, and utilizing gradient low-rank projection, GradNormLoRP significantly reduces memory overhead during training without adding inference burden. We validate its effectiveness both mathematically and empirically. Our experiments demonstrate its efficacy in LLM pre-training and fine-tuning, achieving comparable or superior results to existing PEFT methods.

Acknowledgements

The computational support for this research was provided by the Netherlands Organization for Scientific Research (NWO) under project number EINF-9627.

References

Baevski, A.; Zhou, Y.; Mohamed, A.; and Auli, M. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33: 12449–12460.

Bentivogli, L.; Clark, P.; Dagan, I.; and Giampiccolo, D. 2009. The Fifth PASCAL Recognizing Textual Entailment Challenge. *TAC*, 7(8): 1.

Chen, H.; Raskutti, G.; and Yuan, M. 2019. Non-convex projected gradient descent for generalized low-rank tensor regression. *Journal of Machine Learning Research*, 20(5): 1–37.

Chen, T.; Xu, B.; Zhang, C.; and Guestrin, C. 2016. Training Deep Nets with Sublinear Memory Cost. arXiv:1604.06174.

Chen, Y.; and Wainwright, M. J. 2015. Fast low-rank estimation by projected gradient descent: General statistical and algorithmic guarantees. *arXiv preprint arXiv:1509.03025*.

Dettmers, T.; Lewis, M.; Shleifer, S.; and Zettlemoyer, L. 2022. 8-bit Optimizers via Block-wise Quantization. arXiv:2110.02861.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Frankle, J.; and Carbin, M. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. arXiv:1803.03635.

Frankle, J.; Dziugaite, G. K.; Roy, D.; and Carbin, M. 2020. Linear Mode Connectivity and the Lottery Ticket Hypothesis. In III, H. D.; and Singh, A., eds., *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, 3259–3269.

Gomez, A. N.; Ren, M.; Urtasun, R.; and Grosse, R. B. 2017a. The reversible residual network: Backpropagation without storing activations. *Advances in neural information processing systems*, 30.

Gomez, A. N.; Ren, M.; Urtasun, R.; and Grosse, R. B. 2017b. The Reversible Residual Network: Backpropagation Without Storing Activations. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Gur-Ari, G.; Roberts, D. A.; and Dyer, E. 2018. Gradient Descent Happens in a Tiny Subspace. arXiv:1812.04754.

Hambardzumyan, K.; Khachatrian, H.; and May, J. 2021. Warp: Word-level adversarial reprogramming. *arXiv preprint arXiv:2101.00121*.

He, K.; Chen, X.; Xie, S.; Li, Y.; Dollár, P.; and Girshick, R. 2022. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 16000–16009.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the Knowledge in a Neural Network. arXiv:1503.02531.

Houlsby, N.; Giurgiu, A.; Jastrzebski, S.; Morrone, B.; De Laroussilhe, Q.; Gesmundo, A.; Attariyan, M.; and Gelly, S. 2019. Parameter-efficient transfer learning for NLP. In *International conference on machine learning*, 2790–2799. PMLR.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2022. Lora: Low-rank adaptation of large language models. *International Conference on Learning Representations*.

Kingma, D. P.; and Ba, J. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980.

Kitaev, N.; Kaiser, Ł.; and Levskaya, A. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Kopiczko, D. J.; Blankevoort, T.; and Asano, Y. M. 2024. Vera: Vector-based random matrix adaptation. *International Conference on Learning Representations*.

Koratana, A.; Kang, D.; Bailis, P.; and Zaharia, M. 2019. LIT: Learned Intermediate Representation Training for Model Compression. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 3509–3518. PMLR.

Larsen, B. W.; Fort, S.; Becker, N.; and Ganguli, S. 2022. How many degrees of freedom do we need to train deep networks: a loss landscape perspective. arXiv:2107.05802.

Le Scao, T.; Fan, A.; Akiki, C.; Pavlick, E.; Ilić, S.; Hesslow, D.; Castagné, R.; Luccioni, A. S.; Yvon, F.; Gallé, M.; et al. 2023. Bloom: A 176b-parameter open-access multilingual language model.

Lester, B.; Al-Rfou, R.; and Constant, N. 2021. The power of scale for parameter-efficient prompt tuning. In *EMNLP*.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2019. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. arXiv:1910.13461.

Li, B.; Chen, J.; and Zhu, J. 2023. Memory Efficient Optimizers with 4-bit States. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems*, volume 36, 15136–15171. Curran Associates, Inc.

Li, X. L.; and Liang, P. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Lialin, V.; Muckatira, S.; Shivagunde, N.; and Rumshisky, A. 2023. ReLoRA: High-Rank Training Through Low-Rank Updates. In *Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@ NeurIPS 2023)*.

Liao, B.; Tan, S.; and Monz, C. 2023. Make Pre-trained Model Reversible: From Parameter to Memory Efficient Fine-Tuning. *Advances in Neural Information Processing Systems*, 36.

Liu, H.; Tam, D.; Muqeeth, M.; Mohta, J.; Huang, T.; Bansal, M.; and Raffel, C. A. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35: 1950–1965.

Liu, S.-Y.; Wang, C.-Y.; Yin, H.; Molchanov, P.; Wang, Y.-C. F.; Cheng, K.-T.; and Chen, M.-H. 2024. DoRA: Weight-Decomposed Low-Rank Adaptation. *International Conference on Machine Learning*.

Liu, X.; Zheng, Y.; Du, Z.; Ding, M.; Qian, Y.; Yang, Z.; and Tang, J. 2023. GPT understands, too. *AI Open*.

Liu, Y.; An, C.; and Qiu, X. 2024. Y-tuning: An efficient tuning paradigm for large-scale pre-trained models via label representation learning. *Frontiers of Computer Science*, 18(4): 184320.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Lu, J.; Batra, D.; Parikh, D.; and Lee, S. 2019. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Advances in neural information processing systems*, 32.

Lv, K.; Yang, Y.; Liu, T.; Gao, Q.; Guo, Q.; and Qiu, X. 2023. Full Parameter Fine-tuning for Large Language Models with Limited Resources. arXiv:2306.09782.

Mangalam, K.; Fan, H.; Li, Y.; Wu, C.-Y.; Xiong, B.; Feichtenhofer, C.; and Malik, J. 2022. Reversible Vision Transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10830–10840.

Pfeiffer, J.; Kamath, A.; Rücklé, A.; Cho, K.; and Gurevych, I. 2020. Adapterfusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247*.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020a. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140): 1–67.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020b. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140): 1–67.

Rebuffi, S.-A.; Bilen, H.; and Vedaldi, A. 2017. Learning multiple visual domains with residual adapters. *Advances in neural information processing systems*, 30.

Rücklé, A.; Geigle, G.; Glockner, M.; Beck, T.; Pfeiffer, J.; Reimers, N.; and Gurevych, I. 2020. Adapterdrop: On the efficiency of adapters in transformers. *arXiv preprint arXiv:2010.11918*.

Salimans, T.; and Kingma, D. P. 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29.

Sanh, V.; Debut, L.; Chaumond, J.; and Wolf, T. 2020. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv:1910.01108.

Shazeer, N.; and Stern, M. 2018. Adafactor: Adaptive Learning Rates with Sublinear Memory Cost. In Dy, J.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 4596–4604. PMLR.

Srebro, N.; and Shraibman, A. 2005. Rank, trace-norm and max-norm. In *International conference on computational learning theory*, 545–560. Springer.

Sung, Y.-L.; Cho, J.; and Bansal, M. 2022. Lst: Ladder side-tuning for parameter and memory efficient transfer learning. *Advances in Neural Information Processing Systems*, 35: 12991–13005.

Tan, H.; and Bansal, M. 2019. LXMERT: Learning Cross-Modality Encoder Representations from Transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 5100–5111. Hong Kong, China: Association for Computational Linguistics.

Tay, Y.; Dehghani, M.; Tran, V. Q.; Garcia, X.; Wei, J.; Wang, X.; Chung, H. W.; Bahri, D.; Schuster, T.; Zheng, H. S.; et al. 2023. UL2: Unifying Language Learning Paradigms. In *ICLR*.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. R. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. arXiv:1804.07461.

Xia, W.; Qin, C.; and Hazan, E. 2024. Chain of lora: Efficient fine-tuning of language models via residual learning. *International Conference on Machine Learning*.

Xie, Z.; Zhang, Z.; Cao, Y.; Lin, Y.; Bao, J.; Yao, Z.; Dai, Q.; and Hu, H. 2022. Simmim: A simple framework for masked image modeling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 9653–9663.

Zhang, S.; Roller, S.; Goyal, N.; Artetxe, M.; Chen, M.; Chen, S.; Dewan, C.; Diab, M.; Li, X.; Lin, X. V.; et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Zhao, C.; Liu, S.; Mangalam, K.; Qian, G.; Zohra, F.; Alghannam, A.; Malik, J.; and Ghanem, B. 2024a. Dr2Net: Dynamic Reversible Dual-Residual Networks for Memory-Efficient Finetuning. *Conference on Computer Vision and Pattern Recognition*.

Zhao, J.; Zhang, Z.; Chen, B.; Wang, Z.; Anandkumar, A.; and Tian, Y. 2024b. Galore: Memory-efficient llm training by gradient low-rank projection. *International Conference on Machine Learning*.